START .env.example

```
# Since .env is gitignored, you can use .env.example to build a new `.env` file when you clone the repo.\n# Keep this file up-to-date when you add new variables to `.env`.\n\n# This file will be committed to version control, so make sure not to have any secrets in it.\n# If you are cloning this repo, create a copy of this file named `.env` and populate it with your secrets.\n\n# When adding additional env variables, the schema in /env/schema.mjs should be updated accordingly\n\n# Prisma\nDATABASE_URL=file:./db.sqlite\n\n# Next Auth\n# You can generate the secret via 'openssl rand -base64 32' on Linux\n# More info: https://next-auth.js.org/configuration/options#secret\n# NEXTAUTH_SECRET=\nNEXTAUTH_URL=http://localhost:3000\n\n# Next Auth Github Provider\nGITHUB_ID=\nGITHUB_SECRET=\n
```

END .env.example

START .eslintrc.json

```
{\n    "parser": "@typescript-eslint/parser",\n    "parserOptions": {\n "project": "./tsconfig.json"\n    },\n    "plugins": ["@typescript-eslint"],\n  "extends": ["next/core-web-vitals", "plugin:@typescript-eslint/recommended"],\n    "rules": {\n        "@typescript-eslint/consistent-type-imports": "warn"\n }\n}\n
```

END .eslintrc.json

START .gitignore

```
# See https://help.github.com/articles/ignoring-files/ for more about ignoring files.\n\n# dependencies\n/node_modules\n/.pnp\n.pnp.js\n\n# testing\n/coverage\n\n# database\n/prisma/db.sqlite\n/prisma/db.sqlite-journal\n\n# next.js\n/.next/\n/out/\nnext-env.d.ts\n\n# production\n/build\n\n# misc\n.DS_Store\n*.pem\n\n# debug\nnpm-debug.log*\nyarn-debug.log*\nyarn-error.log*\n.pnpm-debug.log*\n\n# local env files\n# do not commit any .env files to git, except for the .env.example file. https://create.t3.gg/en/usage/env-variables#using-environment-variables\n.env\n.env*.local\n\n# vercel\n.vercel\n\n# typescript\n*.tsbuildinfo\n
```

END .gitignore

START .prettierignore

```
.next\n
```

END .prettierignore

START .sentryclirc

```
[auth]\ntoken=1e73961541e84d1aaf598962bdb6ee97d4ee4bbed5bc49fcb82a7b7edb88b83f
```

END .sentryclirc

START next.config.mjs

```
// @ts-check\nimport { withSentryConfig } from "@sentry/nextjs";\n/**\n * Run `build` or `dev` with `SKIP_ENV_VALIDATION` to skip env validation.\n * This is especially useful for Docker builds.\n */\n!process.env.SKIP_ENV_VALIDATION && (await import("./src/env/server.mjs"));\n\nconst sentryWebpackPluginOptions = {\n    silent: true,\n};\n\n/** @type {import("next").NextConfig} */\nconst config = {\n    reactStrictMode: true,\n    swcMinify: true,\n    i18n: {\n  locales: ["en"],\n        defaultLocale: "en",\n    },\n    sentry: {\n hideSourceMaps: true\n    }\n};\n\nexport default withSentryConfig(config, sentryWebpackPluginOptions);\n
```

END next.config.mjs

START package.json

```
{\n    "name": "os-mock-1",\n    "version": "0.1.0",\n    "private": true,\n
"scripts": {\n        "build": "next build",\n        "dev": "next dev",\n
 "postinstall": "prisma generate",\n        "lint": "next lint",\n
"start": "next start",\n        "format": "prettier --write .",\n
"format:check": "prettier --check ."\n    },\n    "dependencies": {\n
"@chakra-ui/react": "^2.4.3",\n        "@emotion/react": "^11.10.5",\n
"@emotion/styled": "^11.10.5",\n        "@fortawesome/fontawesome-svg-core":
"^6.2.1",\n        "@fortawesome/free-regular-svg-icons": "^6.2.1",\n
"@fortawesome/free-solid-svg-icons": "^6.2.1",\n
"@fortawesome/react-fontawesome": "^0.2.0",\n
"@next-auth/prisma-adapter": "1.0.5",\n        "@prisma/client": "^4.8.1",\n
   "@sentry/nextjs": "^7.26.0",\n        "@tanstack/react-query": "^4.16.0",\n
     "@tanstack/react-table": "^8.7.3",\n        "@tanstack/react-virtual":
"^3.0.0-alpha.0",\n        "@trpc/client": "^10.0.0",\n        "@trpc/next":
"^10.0.0",\n        "@trpc/react-query": "^10.0.0",\n        "@trpc/server":
"^10.0.0",\n        "bcryptjs": "^2.4.3",\n        "clsx": "^1.2.1",\n
"framer-motion": "^7.6.19",\n        "next": "13.0.2",\n        "next-auth":
"4.17.0",\n        "ora": "^6.1.2",\n        "react": "18.2.0",\n
"react-dom": "18.2.0",\n        "superjson": "1.9.1",\n        "zod":
"^3.18.0",\n        "zustand": "^4.1.5"\n    },\n    "devDependencies": {\n
  "@types/node": "^18.0.0",\n        "@types/ora": "^3.2.0",\n
"@types/react": "^18.0.14",\n        "@types/react-dom": "^18.0.5",\n
"@typescript-eslint/eslint-plugin": "^5.33.0",\n
"@typescript-eslint/parser": "^5.33.0",\n        "autoprefixer": "^10.4.7",\n
    "eslint": "^8.26.0",\n        "eslint-config-next": "13.0.2",\n
"postcss": "^8.4.14",\n        "prettier": "^2.7.1",\n
"prettier-plugin-tailwindcss": "^0.1.13",\n        "prisma": "^4.8.1",\n
"tailwindcss": "^3.2.0",\n        "tsx": "^3.12.1",\n        "typescript":
"^4.8.4"\n    },\n    "ct3aMetadata": {\n        "initVersion": "6.11.2"\n
},\n    "prisma": {\n        "seed": "tsx prisma/seed.ts"\n    }\n}\n
```

END package.json

START postcss.config.cjs

```
module.exports = {\n    plugins: {\n        tailwindcss: {},\n
autoprefixer: {},\n    },\n};\n
```

END postcss.config.cjs

START prettier.config.cjs

```
/** @type {import("prettier").Config} */\nmodule.exports = {\n    plugins:
[require.resolve("prettier-plugin-tailwindcss")],\n    tabWidth: 4,\n
printWidth: 120,\n};\n
```

END prettier.config.cjs

START README.md

```
# Create T3 App\n\nThis is a [T3 Stack](https://create.t3.gg/) project
bootstrapped with `create-t3-app`.\n\n## What's next? How do I make an app with
this?\n\nWe try to keep this project as simple as possible, so you can start
with just the scaffolding we set up for you, and add additional things later
when they become necessary.\n\nIf you are not familiar with the different
technologies used in this project, please refer to the respective docs. If you
still are in the wind, please join our [Discord](https://t3.gg/discord) and ask
for help.\n\n-   [Next.js](https://nextjs.org)\n-
[NextAuth.js](https://next-auth.js.org)\n-   [Prisma](https://prisma.io)\n-
[Tailwind CSS](https://tailwindcss.com)\n-   [tRPC](https://trpc.io)\n\n## Learn
More\n\nTo learn more about the [T3 Stack](https://create.t3.gg/), take a look
at the following resources:\n\n-   [Documentation](https://create.t3.gg/)\n-
```

```
[Learn the T3
Stack](https://create.t3.gg/en/faq#what-learning-resources-are-currently-availab
le) \u2014 Check out these awesome tutorials\n\nYou can check out the
[create-t3-app GitHub repository](https://github.com/t3-oss/create-t3-app)
\u2014 your feedback and contributions are welcome!\n\n## How do I deploy
this?\n\nFollow our deployment guides for
[Vercel](https://create.t3.gg/en/deployment/vercel) and
[Docker](https://create.t3.gg/en/deployment/docker) for more information.\n
```

END README.md

START sentry.client.config.ts

```
import * as Sentry from "@sentry/nextjs";\n\nSentry.init({\n    dsn:
"https://10151740151f4d418ea2c98b8480231d@sentry.danielraybone.com/4",\n  \n
// Set tracesSampleRate to 1.0 to capture 100%\n    // of transactions for
performance monitoring.\n    // We recommend adjusting this value in
production\n    tracesSampleRate: 1.0,\n  });
```

END sentry.client.config.ts

START sentry.properties

```
defaults.url=https://sentry.danielraybone.com/\ndefaults.org=secureheaven\ndefau
lts.project=os-mock-1
```

END sentry.properties

START sentry.server.config.ts

```
import * as Sentry from "@sentry/nextjs";\n\nSentry.init({\n    dsn:
"https://10151740151f4d418ea2c98b8480231d@sentry.danielraybone.com/4",\n  \n
// Set tracesSampleRate to 1.0 to capture 100%\n    // of transactions for
performance monitoring.\n    // We recommend adjusting this value in
production\n    tracesSampleRate: 1.0,\n  });
```

END sentry.server.config.ts

START tailwind.config.cjs

```
/** @type {import('tailwindcss').Config} */\nmodule.exports = {\n    content:
["./src/**/*.{js,ts,jsx,tsx}"],\n    theme: {\n        extend: {},\n    },\n
plugins: [],\n};\n
```

END tailwind.config.cjs

START tsconfig.json

```
{\n    "compilerOptions": {\n        "target": "es2017",\n        "lib": ["dom",
"dom.iterable", "esnext"],\n        "allowJs": true,\n        "skipLibCheck":
true,\n        "strict": true,\n        "forceConsistentCasingInFileNames":
true,\n        "noEmit": true,\n        "esModuleInterop": true,\n
"module": "esnext",\n        "moduleResolution": "node",\n
"resolveJsonModule": true,\n        "isolatedModules": true,\n        "jsx":
"preserve",\n        "incremental": true,\n        "noUncheckedIndexedAccess":
true,\n        "baseUrl": "src",\n        "paths": {\n
"@components/*": ["components/*"],\n            "@hooks/*": ["hooks/*"],\n
    "@schema/*": ["schema/*"],\n            "@server/*": ["server/*"],\n
    "@types/*": ["types/*"],\n            "@utils/*": ["utils/*"]\n        }\n
 },\n    "include": ["next-env.d.ts", "**/*.ts", "**/*.tsx", "**/*.cjs",
"**/*.mjs"],\n    "exclude": ["node_modules"]\n}\n
```

END tsconfig.json

START schema.prisma

```
// This is your Prisma schema file,\n// learn more about it in the docs: https://pris.ly/d/prisma-schema\n\ngenerator client {\n  provider = "prisma-client-js"\n}\n\ndatasource db {\n  provider = "sqlite"\n  // NOTE: When using postgresql, mysql or sqlserver, uncomment the @db.Text annotations in model Account below\n  // Further reading:\n  // https://next-auth.js.org/adapters/prisma#create-the-prisma-schema\n  // https://www.prisma.io/docs/reference/api-reference/prisma-schema-reference#string\n  url      = env("DATABASE_URL")\n}\n\n// Necessary for Next auth\nmodel Account {\n  id                 String  @id @default(cuid())\n  userId             String\n  type               String\n  provider           String\n  providerAccountId  String\n  refresh_token      String?\n  access_token       String?\n  refresh_token_expires_in Int?\n  expires_at         Int?\n  token_type         String?\n  scope              String?\n  id_token           String?\n  session_state      String?\n  user               User    @relation(fields: [userId], references: [id], onDelete: Cascade)\n\n  @@unique([provider, providerAccountId])\n}\n\nmodel Session {\n  id           String   @id @default(cuid())\n  sessionToken String   @unique\n  userId       String\n  expires      DateTime\n  user         User     @relation(fields: [userId], references: [id], onDelete: Cascade)\n}\n\nmodel User {\n  id            String     @id @default(cuid())\n  name          String?\n  email         String?    @unique\n  emailVerified DateTime?\n  image         String?\n  accounts      Account[]\n  sessions      Session[]\n  courses       CourseUser[]\n}\n\nmodel VerificationToken {\n  identifier String\n  token      String   @unique\n  expires    DateTime\n\n  @@unique([identifier, token])\n}\n\nmodel Course {\n  id            String       @id @default(cuid())\n  name          String\n  desc          String\n  published     Boolean      @default(false)\n  createdAt     DateTime     @default(now())\n  updatedAt     DateTime     @updatedAt\n  enrolledUsers CourseUser[]\n  assignments   Assignment[]\n}\n\nmodel Assignment {\n  id        String     @id @default(cuid())\n  name      String\n  dueDate   DateTime\n  published Boolean    @default(false)\n  createdAt DateTime   @default(now())\n  updatedAt DateTime   @updatedAt\n  course    Course     @relation(fields: [courseId], references: [id])\n  documents Document[]\n\n  courseId String\n}\n\nmodel Document {\n  id           String      @id @default(cuid())\n  createdAt    DateTime    @default(now())\n  updatedAt    DateTime    @updatedAt\n  Assignment   Assignment? @relation(fields: [assignmentId], references: [id])\n  assignmentId String?\n}\n\nmodel CourseUser {\n  course    Course   @relation(fields: [courseId], references: [id])\n  user      User     @relation(fields: [userId], references: [id])\n  courseId  String\n  userId    String\n  createdAt DateTime @default(now())\n\n  @@unique([courseId, userId])\n}\n\nmodel Role {\n  id     String  @id @default(cuid())\n  name   String\n  active Boolean @default(true)\n\n  createdAt  DateTime     @default(now())\n  updatedAt  DateTime     @updatedAt\n  Permssions Permssions[]\n}\n\nmodel Permssions {\n  id   String @id @default(cuid())\n  role Role   @relation(fields: [roleId], references: [id])\n\n  roleId String\n}\n
```

END schema.prisma

START seed.ts

```
import { type Course, PrismaClient } from "@prisma/client";\nimport ora, { type Ora } from "ora";\n\nconst prisma = new PrismaClient();\n\n(async () => {\n  let spinner: Ora;\n    const user = await prisma.user.findFirst();\n\n    if (!user) {\n      throw new Error("User does not exist, login using oauth first before seeding the database.");\n    }\n\n    const courses: Omit<Course, "id">[] = [\n      {\n        createdAt: new Date(),\n        desc: "New T-Level Digital Production & Design course",\n        name: "T-Level Digital Production & Design",\n        published: true,\n        updatedAt: new Date(),\n      },\n      {\n        createdAt: new Date(),\n        desc: "BTEC Business course",\n        name: "BTEC Business",\n        published: false,\n        updatedAt: new Date(),\n      },\n      {\n        createdAt: new Date(),\n        desc: "Computer Science learning course",\n        name: "Computer Science",\n        published: true,\n        updatedAt: new Date(),\n      },\n      {\n        createdAt: new Date(),\n        desc: "cook food nerds",\n
```

```
        name: "Cooking",\n            published: true,\n            updatedAt:
new Date(),\n        },\n    ];\n\n    for (const course of courses) {\n
const exists = await prisma.course.findFirst({\n            where: {\n
     name: {\n                    equals: course.name,\n                },\n
      },\n        });\n\n        if (exists) {\n
console.warn(`${course.name} is already added within the database,
skipping...`);\n            continue;\n        }\n\n        spinner =
ora(`Creating course ${course.name}`).start();\n\n        const created = await
prisma.course.create({\n            data: course,\n        });\n\n
spinner.succeed(`Created course ${course.name} (${created.id})`);\n\n
spinner = ora(`Adding ${user.name} (${user.id}) to ${course.name}
(${created.id})`).start();\n\n        await prisma.courseUser.create({\n
   data: {\n                course: {\n                    connect: {\n
             id: created.id,\n                    },\n                },\n
          user: {\n                    connect: {\n                        id:
user.id,\n                    },\n                },\n            },\n
});\n\n        spinner.succeed(`Added ${user.name} (${user.id}) to
${course.name} (${created.id})`);\n\n        const numberOfAssignments =
100;\n\n        spinner = ora(\n            `Creating ${numberOfAssignments}
assignments for course ${course.name} (${created.id}). This might take a
moment`\n        ).start();\n\n        await prisma.$transaction(\n
async () => {\n                const inserts = [];\n                for (let i =
0; i < numberOfAssignments; i++) {\n                    inserts.push(\n
             await prisma.assignment.create({\n
data: {\n                                dueDate: new Date(),\n
             name: `Test assignment #${i}`,\n
course: { connect: { id: created.id } },\n                        },\n
                })\n                    );\n                }\n\n
 return inserts;\n            },\n            {\n                // Give each
assignment 3 seconds to complete\n                timeout: numberOfAssignments *
1000 * 3,\n            }\n        );\n\n        spinner.succeed(`Created
${numberOfAssignments} assignments for course ${course.name}
(${created.id})`);\n    }\n})();\n
```

END seed.ts

START autoTable.tsx

```
import type { RefObject } from "react";\nimport { flexRender, type Row, type
Table as TableType } from "@tanstack/react-table";\nimport { useVirtual } from
"@tanstack/react-virtual";\nimport { Table, Thead, Tbody, Tr, Th, Td, Tfoot }
from "@chakra-ui/react";\nimport Spinner from "./spinner/spinner";\n\ntype
TableProps<T> = {\n    table: TableType<T>;\n    refReq:
RefObject<HTMLDivElement>;\n    isFetching?: boolean;\n};\n\nconst AutoTable =
<T,>({ table, refReq, isFetching }: TableProps<T>) => {\n    const { rows } =
table.getRowModel();\n\n    const rowVirtualizer = useVirtual({\n
parentRef: refReq,\n        size: rows.length,\n        overscan: 10,\n
});\n\n    const { virtualItems: virtualRows, totalSize } = rowVirtualizer;\n
const paddingTop = virtualRows.length > 0 ? virtualRows?.[0]?.start || 0 : 0;\n
  const paddingBottom = virtualRows.length > 0 ? totalSize -
(virtualRows?.[virtualRows.length - 1]?.end || 0) : 0;\n\n    return (\n
<>\n            <Table>\n                <Thead>\n
{table.getHeaderGroups().map((headerGroup) => (\n                        <Tr
key={headerGroup.id}>\n
{headerGroup.headers.map((header) => {\n                            return
(\n                                <Th key={header.id}
colSpan={header.colSpan} style={{ width: header.getSize() }}>\n
                    {header.isPlaceholder ? null : (\n
                <div\n                                    {...{\n
                                        className:
header.column.getCanSort()\n
   ? "cursor-pointer select-none"\n
          : "",\n                                    onClick:
header.column.getToggleSortingHandler(),\n
         }}\n                                    >\n
                    {flexRender(header.column.columnDef.header,
header.getContext())}\n                                    {{\n
                                        asc: " \U0001f53c",\n
```

```
                                    desc: " \U0001f53d",\n
                                }[header.column.getIsSorted() as string] ?? null}\n
                                    </div>\n
        )}\n                                        </Th>\n
      );\n                                  })}\n                                  </Tr>\n
            ))}\n                       </Thead>\n                    <Tbody>\n
        {paddingTop > 0 && (\n                          <Tr>\n
       <Td style={{ height: `${paddingTop}px` }} />\n
</Tr>\n                      )}\n                      {virtualRows.map((virtualRow)
=> {\n                         const row = rows[virtualRow.index] as Row<T>;\n
                    return (\n                                  <Tr key={row.id}>\n
                            {row.getVisibleCells().map((cell) => {\n
                       return (\n                                                  <Td
key={cell.id}>\n
{flexRender(cell.column.columnDef.cell, cell.getContext())}\n
                    </Td>\n                                              );\n
                  })}\n                                 </Tr>\n
 );\n                    })}\n                       {paddingBottom > 0 && (\n
                    <Tr>\n                                <Td style={{ height:
`${paddingBottom}px` }} />\n                          </Tr>\n
)}\n                  </Tbody>\n              </Table>\n            {isFetching &&
(\n                 <div className="flex w-full justify-center ">\n
     <Spinner />\n                  </div>\n             )}\n        </>\n
);\n};\n\nexport default AutoTable;\n
```

END autoTable.tsx

START courseBox.tsx

```
import { Card, CardHeader, Heading, CardBody, Text, CardFooter, Button } from
"@chakra-ui/react";\nimport type { Course } from "@prisma/client";\nimport {
useRouter } from "next/router";\n\ntype UnneedKeys = "name" | "desc" |
"id";\ntype CourseBoxProps = Pick<Course, UnneedKeys>;\n\nconst CourseBox:
React.FC<CourseBoxProps> = ({ id, name, desc }) => {\n    const router =
useRouter();\n    return (\n        <Card>\n            <CardHeader>\n
     <Heading size="md">{name}</Heading>\n            </CardHeader>\n
<CardBody>\n                <Text>{desc}</Text>\n            </CardBody>\n
     <CardFooter>\n                  <Button onClick={() =>
router.push(`/course/${id}`)}>View</Button>\n            </CardFooter>\n
</Card>\n    );\n};\n\nexport default CourseBox;\n
```

END courseBox.tsx

START layout.tsx

```
import { Breadcrumb, BreadcrumbItem as ChakraBreadcrumbItem, BreadcrumbLink,
Container } from "@chakra-ui/react";\nimport useBreadcrumbsStore, {
BreadcrumbItem } from "@hooks/useBreadcrumbsStore";\nimport { capital } from
"@utils/stringFormat";\nimport { signIn, useSession } from
"next-auth/react";\nimport { type ReactNode, useEffect } from "react";\nimport
Navigation from "./nav";\n\ntype LayoutProps = {\n    children: ReactNode |
ReactNode[];\n};\n\nconst BreadcrumbItem: React.FC<BreadcrumbItem> = ({ href,
name, ...props }) => (\n    <ChakraBreadcrumbItem {...props}>\n
<BreadcrumbLink href={href}>{capital(name)}</BreadcrumbLink>\n
</ChakraBreadcrumbItem>\n);\n\nconst Layout: React.FC<LayoutProps> = ({ children
}) => {\n    const breadcrumbs = useBreadcrumbsStore((state) => state.items);\n
  const session = useSession();\n\n    useEffect(() => {\n        if
(session.status === "unauthenticated") {\n            signIn("github");\n
}\n    }, [session.status]);\n\n    return (\n        <div className="flex
flex-col">\n            <Navigation />\n            <Breadcrumb
className="p-4">\n                <BreadcrumbItem href="/" name="Home" />\n
          {breadcrumbs.map(({ href, name }) => (\n
<BreadcrumbItem key={href} href={href} name={name} />\n                ))}\n
       </Breadcrumb>\n            <Container maxW="2xl">{children}</Container>\n
       </div>\n    );\n};\n\nexport default Layout;\n
```

END layout.tsx

START logo.tsx

```
import type { DetailedHTMLProps, HTMLAttributes } from "react";\n\ntype
LogoProps = {\n    width: string;\n    height: string;\n} &
DetailedHTMLProps<HTMLAttributes<HTMLDivElement>, HTMLDivElement>;\n\nconst
Logo: React.FC<LogoProps> = ({ width, height, ...props }) => (\n    <div
{...props}>\n        <svg xmlns="http://www.w3.org/2000/svg" width={width}
height={height} viewBox="0 0 32 32">\n            <g transform="matrix(.05696 0
0 .05696 .647744 2.43826)" fill="none" fill-rule="evenodd">\n
<circle r="50.167" cy="237.628" cx="269.529" fill="#00d8ff" />\n
<g stroke="#00d8ff" stroke-width="24">\n                    <path d="M269.53
135.628c67.356 0 129.928 9.665 177.107 25.907 56.844 19.57 91.794 49.233 91.794
76.093 0 27.99-37.04 59.503-98.083 79.728-46.15 15.29-106.88 23.272-170.818
23.272-65.554 0-127.63-7.492-174.3-23.44-59.046-20.182-94.61-52.103-94.61-79.56
0-26.642 33.37-56.076 89.415-75.616 47.355-16.51 111.472-26.384 179.486-26.384z"
/>\n                    <path d="M180.736 186.922c33.65-58.348 73.28-107.724
110.92-140.48C337.006 6.976 380.163-8.48 403.43 4.937c24.248 13.983 33.042
61.814 20.067 124.796-9.8 47.618-33.234 104.212-65.176 159.6-32.75 56.788-70.25
106.82-107.377 139.272-46.98 41.068-92.4 55.93-116.185
42.213-23.08-13.3-31.906-56.92-20.834-115.233 9.355-49.27 32.832-109.745
66.8-168.664z" />\n                    <path d="M180.82 289.482C147.075 231.2
124.1 172.195 114.51 123.227c-11.544-59-3.382-104.11 19.864-117.566
24.224-14.024 70.055 2.244 118.14 44.94 36.356 32.28 73.688 80.837 105.723
136.173 32.844 56.733 57.46 114.21 67.036 162.582 12.117 61.213 2.31
107.984-21.453 121.74-23.057
13.348-65.25-.784-110.24-39.5-38.013-32.71-78.682-83.253-112.76-142.115z" />\n
            </g>\n            </g>\n        </svg>\n    </div>\n);\n\nexport
default Logo;\n
```

END logo.tsx

START nav.tsx

```
import { signIn, signOut, useSession } from "next-auth/react";\nimport Logo from
"./logo";\n\nconst Navigation: React.FC = () => {\n    const { status } =
useSession();\n\n    return (\n        <div className="flex items-center
justify-between bg-zinc-900 p-4">\n            <Logo height="40" width="40" />\n
            <div className="">\n                <div>\n
{status === "authenticated" ? (\n                        <button onClick={() =>
signOut()}>Signout</button>\n                    ) : (\n
<button onClick={() => signIn("github")}>Login</button>\n
)}\n                </div>\n            </div>\n        </div>\n
);\n};\n\nexport default Navigation;\n
```

END nav.tsx

START table.tsx

```
import { useRef } from "react";\nimport { flexRender, type Row, type Table }
from "@tanstack/react-table";\nimport { Table as ChakraTable, Thead, Tbody,
Tfoot, Tr, Th, Td } from "@chakra-ui/react";\nimport { useVirtual } from
"@tanstack/react-virtual";\n\ntype TableProps = { table: Table<unknown>
};\n\nconst AutoTable: React.FC<TableProps> = ({ table }) => {\n    const
tableContainerRef = useRef<HTMLDivElement>(null);\n\n    const { rows } =
table.getRowModel();\n\n    //Virtualizing is optional, but might be necessary
if we are going to potentially have hundreds or thousands of rows\n    const
rowVirtualizer = useVirtual({\n        parentRef: tableContainerRef,\n
size: rows.length,\n        overscan: 10,\n    });\n    const { virtualItems:
virtualRows, totalSize } = rowVirtualizer;\n    const paddingTop =
virtualRows.length > 0 ? virtualRows?.[0]?.start || 0 : 0;\n    const
paddingBottom = virtualRows.length > 0 ? totalSize -
(virtualRows?.[virtualRows.length - 1]?.end || 0) : 0;\n\n    return (\n
<ChakraTable>\n            <Thead>\n
{table.getHeaderGroups().map((headerGroup) => (\n                    <Tr
key={headerGroup.id}>\n                        {headerGroup.headers.map((header)
=> (\n                            <Th key={header.id}>\n
      {header.isPlaceholder\n                                ? null\n
```

```
                            : flexRender(header.column.columnDef.header,
header.getContext())}\n                                    </Th>\n
 ))}\n                      </Tr>\n                    ))}\n            </Thead>\n
       <Tbody>\n                  {paddingTop > 0 && (\n                      <tr>\n
                      <td style={{ height: `${paddingTop}px` }} />\n
         </tr>\n                  )}\n                  {virtualRows.map((virtualRow)
=> {\n                    const row = rows[virtualRow.index] as Row<unknown>;\n
                  return (\n                           <tr key={row.id}>\n
                  {row.getVisibleCells().map((cell) => {\n
          return (\n                                  <td
key={cell.id}>{flexRender(cell.column.columnDef.cell, cell.getContext())}</td>\n
                              );\n                              })}\n
            </tr>\n                    );\n                })}\n
{paddingBottom > 0 && (\n                      <tr>\n                          <td
style={{ height: `${paddingBottom}px` }} />\n                      </tr>\n
       )}\n            </Tbody>\n            <Tfoot>\n
{table.getFooterGroups().map((footerGroup) => (\n                      <Tr
key={footerGroup.id}>\n                        {footerGroup.headers.map((header)
=> (\n                          <Th key={header.id}>\n
       {header.isPlaceholder\n                                    ? null\n
                            : flexRender(header.column.columnDef.footer,
header.getContext())}\n                                    </Th>\n
 ))}\n                      </Tr>\n                  ))}\n            </Tfoot>\n
   </ChakraTable>\n    );\n};\n\nexport default AutoTable;\n
```

END table.tsx

START spinner.module.css

```
.ldsEllipsis {\n    display: inline-block;\n    position: relative;\n    width:
80px;\n    height: 80px;\n}\n.ldsEllipsis div {\n    position: absolute;\n
top: 33px;\n    width: 13px;\n    height: 13px;\n    border-radius: 50%;\n
background: #fff;\n    animation-timing-function: cubic-bezier(0, 1, 1,
0);\n}\n.ldsEllipsis div:nth-child(1) {\n    left: 8px;\n    animation:
ldsEllipsis1 0.6s infinite;\n}\n.ldsEllipsis div:nth-child(2) {\n    left:
8px;\n    animation: ldsEllipsis2 0.6s infinite;\n}\n.ldsEllipsis
div:nth-child(3) {\n    left: 32px;\n    animation: ldsEllipsis2 0.6s
infinite;\n}\n.ldsEllipsis div:nth-child(4) {\n    left: 56px;\n    animation:
ldsEllipsis3 0.6s infinite;\n}\n@keyframes ldsEllipsis1 {\n    0% {\n
transform: scale(0);\n    }\n    100% {\n        transform: scale(1);\n
}\n}\n@keyframes ldsEllipsis3 {\n    0% {\n        transform: scale(1);\n    }\n
   100% {\n        transform: scale(0);\n    }\n}\n@keyframes ldsEllipsis2 {\n
 0% {\n        transform: translate(0, 0);\n    }\n    100% {\n
transform: translate(24px, 0);\n    }\n}\n
```

END spinner.module.css

START spinner.tsx

```
import style from "./spinner.module.css";\nimport clsx from "clsx";\n\ntype
SpinnerProps = {\n    fullScreen?: boolean;\n    className?:
string;\n};\n\nconst Spinner: React.FC<SpinnerProps> = ({ fullScreen = false,
className }) => (\n    <div className={clsx(className, { "flex h-screen w-screen
items-center justify-center": fullScreen })}>\n        <div
className={style.ldsEllipsis}>\n            <div></div>\n
<div></div>\n            <div></div>\n            <div></div>\n        </div>\n
  </div>\n);\n\nexport default Spinner;\n
```

END spinner.tsx

START client.mjs

```
// @ts-check\nimport { clientEnv, clientSchema } from "./schema.mjs";\n\nconst
_clientEnv = clientSchema.safeParse(clientEnv);\n\nexport const formatErrors =
(\n    /** @type {import('zod').ZodFormattedError<Map<string,string>,string>}
*/\n    errors\n) =>\n    Object.entries(errors)\n        .map(([name, value])
=> {\n            if (value && "_errors" in value) return `${name}:
```

```
${value._errors.join(", ")}\\n`;\n        })\n        .filter(Boolean);\n\nif
(!_clientEnv.success) {\n    console.error("\u274c Invalid environment
variables:\\n", ...formatErrors(_clientEnv.error.format()));\n    throw new
Error("Invalid environment variables");\n}\n\nfor (let key of
Object.keys(_clientEnv.data)) {\n    if (!key.startsWith("NEXT_PUBLIC_")) {\n
    console.warn(`\u274c Invalid public environment variable name: ${key}. It
must begin with 'NEXT_PUBLIC_'`);\n\n        throw new Error("Invalid public
environment variable name");\n    }\n}\n\nexport const env = _clientEnv.data;\n
```

END client.mjs

START schema.mjs

```
// @ts-check\nimport { z } from "zod";\n\n/**\n * Specify your server-side
environment variables schema here.\n * This way you can ensure the app isn't
built with invalid env vars.\n */\nexport const serverSchema = z.object({\n
DATABASE_URL: z.string().url(),\n    NODE_ENV: z.enum(["development", "test",
"production"]),\n    NEXTAUTH_SECRET: process.env.NODE_ENV === "production" ?
z.string().min(1) : z.string().min(1).optional(),\n    NEXTAUTH_URL:
z.preprocess(\n        // This makes Vercel deployments not fail if you don't
set NEXTAUTH_URL\n        // Since NextAuth.js automatically uses the VERCEL_URL
if present.\n        (str) => process.env.VERCEL_URL ?? str,\n        //
VERCEL_URL doesn't include `https` so it cant be validated as a URL\n
process.env.VERCEL ? z.string() : z.string().url()\n    ),\n    GITHUB_ID:
z.string(),\n    GITHUB_SECRET: z.string(),\n});\n\n/**\n * Specify your
client-side environment variables schema here.\n * This way you can ensure the
app isn't built with invalid env vars.\n * To expose them to the client, prefix
them with `NEXT_PUBLIC_`.\n */\nexport const clientSchema = z.object({\n    //
NEXT_PUBLIC_CLIENTVAR: z.string(),\n});\n\n/**\n * You can't destruct
`process.env` as a regular object, so you have to do\n * it manually here. This
is because Next.js evaluates this at build time,\n * and only used environment
variables are included in the build.\n * @type {{ [k in keyof z.infer<typeof
clientSchema>]: z.infer<typeof clientSchema>[k] | undefined }}\n */\nexport
const clientEnv = {\n    // NEXT_PUBLIC_CLIENTVAR:
process.env.NEXT_PUBLIC_CLIENTVAR,\n};\n
```

END schema.mjs

START server.mjs

```
// @ts-check\n/**\n * This file is included in `/next.config.mjs` which ensures
the app isn't built with invalid env vars.\n * It has to be a `.mjs`-file to be
imported there.\n */\nimport { serverSchema } from "./schema.mjs";\nimport { env
as clientEnv, formatErrors } from "./client.mjs";\n\nconst _serverEnv =
serverSchema.safeParse(process.env);\n\nif (!_serverEnv.success) {\n
console.error("\u274c Invalid environment variables:\\n",
...formatErrors(_serverEnv.error.format()));\n    throw new Error("Invalid
environment variables");\n}\n\nfor (let key of Object.keys(_serverEnv.data)) {\n
   if (key.startsWith("NEXT_PUBLIC_")) {\n        console.warn("\u274c You are
exposing a server-side env-variable:", key);\n\n        throw new Error("You are
exposing a server-side env-variable");\n    }\n}\n\nexport const env = {
..._serverEnv.data, ...clientEnv };\n
```

END server.mjs

START useAssignmentsFromCourse.tsx

```
import { trpc } from "@utils/trpc";\n\nconst useCourse = (courseId: string) =>
{\n    return trpc.courses.get.useQuery(courseId, { enabled: !!courseId
});\n};\n\nexport default useCourse;\n
```

END useAssignmentsFromCourse.tsx

START useBool.tsx

```
import { useState } from "react";\n\nconst useBool = (initalValue: boolean):
```

```
[boolean, () => void] => {\n    const [value, setValue] =
useState(initalValue);\n\n    const toggle = () => setValue((prev) =>
!prev);\n\n    return [value, toggle];\n};\n\nexport default useBool;\n
```

END useBool.tsx

START useBreadcrumb.tsx

```
import { useEffect } from "react";\nimport useBreadcrumbsStore, { type
BreadcrumbItem } from "./useBreadcrumbsStore";\n\nconst useBreadcrumb = (item:
BreadcrumbItem) => {\n    const add = useBreadcrumbsStore((state) =>
state.add);\n    const remove = useBreadcrumbsStore((state) =>
state.remove);\n\n    useEffect(() => {\n        add(item);\n        return ()
=> {\n            remove();\n        };\n    }, [add, remove,
item]);\n};\n\nexport default useBreadcrumb;\n
```

END useBreadcrumb.tsx

START useBreadcrumbsStore.tsx

```
import create from "zustand";\n\nexport type BreadcrumbItem = {\n    name:
string;\n    href: string;\n};\n\ntype BreadcrumbState = {\n    items:
BreadcrumbItem[];\n    add: (item: BreadcrumbItem) => void;\n    remove: () =>
void;\n};\n\nconst useBreadcrumbsStore = create<BreadcrumbState>((set) => ({\n
 items: [],\n    add: (item: BreadcrumbItem) => set((state) => ({ items:
[...state.items, item] })),\n    remove: () => set((state) => ({ items:
state.items.slice(0, state.items.length - 1) })),\n}));\n\nexport default
useBreadcrumbsStore;\n
```

END useBreadcrumbsStore.tsx

START useCourse.tsx

```
import { trpc } from "@utils/trpc";\n\nconst useCourse = (courseId: string) =>
{\n    return trpc.courses.get.useQuery(courseId, { enabled: !!courseId
});\n};\n\nexport default useCourse;\n
```

END useCourse.tsx

START useCourseFromRoute.tsx

```
import useCourse from "./useCourse";\nimport useRouteParam from
"./useRouteParam";\n\nconst useCourseFromRoute = () => {\n    const courseId =
useRouteParam("id");\n    return useCourse(courseId);\n};\n\nexport default
useCourseFromRoute;\n
```

END useCourseFromRoute.tsx

START useGetCourses.tsx

```
import type { CourseFilter } from "@schema/courseFilterSchema";\nimport { trpc }
from "@utils/trpc";\n\nconst useCourses = (filter: Partial<CourseFilter>) => {\n
   const defaults: CourseFilter = {\n        all: false,\n        amount: 100,\n
   };\n    return trpc.courses.enrolled.useQuery({ ...defaults, ...filter
});\n};\n\nexport default useCourses;\n
```

END useGetCourses.tsx

START useGetUser.tsx

```
import type { User } from "@prisma/client";\nimport { trpc } from
"@utils/trpc";\n\nconst useGetUsers = () => {\n    const { data, ...rest } =
trpc.users.all.useQuery(undefined, { placeholderData: [] });\n\n    return {
data: (data ?? []) as User[], ...rest };\n};\n\nexport default useGetUsers;\n
```

END useGetUser.tsx

START useRouteParam.tsx

```
import { useEffect } from "react";\nimport { useState } from "react";\nimport {
useRouter } from "next/router";\n\nconst useRouteParam = (param: string) => {\n
  const router = useRouter();\n\n    const [value, setValue] = useState("");\n\n
   useEffect(() => {\n        const id = router.query[param];\n        if
(typeof id === "object") {\n            setValue(id[0] ?? "");\n
return;\n        }\n        setValue(id ?? "");\n    }, [router, param]);\n\n
return value;\n};\n\nexport default useRouteParam;\n
```

END useRouteParam.tsx

START courses.tsx

```
import { Heading, SimpleGrid, Text } from "@chakra-ui/react";\nimport CourseBox
from "@components/courseBox";\nimport Layout from "@components/layout";\nimport
useCourses from "@hooks/useGetCourses";\n\nconst Courses = () => {\n    const {
data } = useCourses({\n        all: true,\n    });\n\n    return (\n
<Layout>\n            <Heading as="h1" size="4xl">\n                Your
Courses\n            </Heading>\n            <Text fontSize="3xl">Check out your
current courses!</Text>\n            <SimpleGrid spacing={4}
templateColumns="repeat(auto-fill, minmax(200px, 1fr))">\n
{data?.map((course) => (\n                    <CourseBox key={course.id}
{...course} />\n                ))}\n            </SimpleGrid>\n
</Layout>\n    );\n};\n\nexport default Courses;\n
```

END courses.tsx

START dashboard.tsx

```
import { Button, Heading, SimpleGrid, Text } from "@chakra-ui/react";\nimport
CourseBox from "@components/courseBox";\nimport Layout from
"@components/layout";\nimport useCourses from "@hooks/useGetCourses";\nimport {
capital } from "@utils/stringFormat";\nimport { useSession } from
"next-auth/react";\nimport { useRouter } from "next/router";\n\nconst Dashboard
= () => {\n    const router = useRouter();\n    const { data: sessionData } =
useSession();\n    const { data } = useCourses({\n        all: true,\n
amount: 5,\n    });\n\n    return (\n        <Layout>\n            <Heading
as="h1" size="4xl">\n                Welcome back,
{capital(sessionData?.user?.name)}!\n            </Heading>\n            <div
className="m-4 flex justify-between">\n                <Text
fontSize="3xl">Recent Courses:</Text>\n                <Button onClick={() =>
router.push("/courses")}>View All</Button>\n            </div>\n\n
<SimpleGrid spacing={4} templateColumns="repeat(auto-fill, minmax(200px,
1fr))">\n                {data?.map((course) => (\n
<CourseBox key={course.id} {...course} />\n                ))}\n
</SimpleGrid>\n        </Layout>\n    );\n};\n\nexport default Dashboard;\n
```

END dashboard.tsx

START index.tsx

```
import { type NextPage } from "next";\nimport Head from "next/head";\nimport
Link from "next/link";\nimport { signIn, signOut, useSession } from
"next-auth/react";\n\nimport { trpc } from "@utils/trpc";\n\nconst Home:
NextPage = () => {\n    // const hello = trpc.example.hello.useQuery({ text:
"from tRPC" });\n\n    return (\n        <>\n            <Head>\n
<title>Create T3 App</title>\n                <meta name="description"
content="Generated by create-t3-app" />\n                <link rel="icon"
href="/favicon.ico" />\n            </Head>\n            <main className="flex
min-h-screen flex-col items-center justify-center bg-gradient-to-b
from-[#2e026d] to-[#15162c]">\n                <div className="container flex
flex-col items-center justify-center gap-12 px-4 py-16 ">\n
<h1 className="text-5xl font-extrabold tracking-tight text-white
```

```
sm:text-[5rem]">\n                          Create <span
className="text-[hsl(280,100%,70%)]">T3</span> App\n                    </h1>\n
                <div className="grid grid-cols-1 gap-4 sm:grid-cols-2
md:gap-8">\n                          <Link\n
className="flex max-w-xs flex-col gap-4 rounded-xl bg-white/10 p-4 text-white
hover:bg-white/20"\n
href="https://create.t3.gg/en/usage/first-steps"\n
target="_blank"\n                          >\n                              <h3
className="text-2xl font-bold">First Steps \u2192</h3>\n
   <div className="text-lg">\n                                  Just the basics -
Everything you need to know to set up your database and\n
        authentication.\n                              </div>\n
    </Link>\n                        <Link\n
className="flex max-w-xs flex-col gap-4 rounded-xl bg-white/10 p-4 text-white
hover:bg-white/20"\n
href="https://create.t3.gg/en/introduction"\n
target="_blank"\n                          >\n                              <h3
className="text-2xl font-bold">Documentation \u2192</h3>\n
     <div className="text-lg">\n                                Learn more about
Create T3 App, the libraries it uses, and how to deploy it.\n
        </div>\n                        </Link>\n                    </div>\n
                <div className="flex flex-col items-center gap-2">\n
          <p className="text-2xl text-white">\n                              {/*
{hello.data ? hello.data.greeting : "Loading tRPC query..."} */}\n
         </p>\n                        <AuthShowcase />\n
</div>\n                </div>\n            </main>\n        </>\n
);\n};\n\nexport default Home;\n\nconst AuthShowcase: React.FC = () => {\n
const { data: sessionData } = useSession();\n\n    const { data: secretMessage }
= trpc.auth.getSecretMessage.useQuery(\n        undefined, // no input\n
{ enabled: sessionData?.user !== undefined }\n    );\n\n    return (\n
<div className="flex flex-col items-center justify-center gap-4">\n
<p className="text-center text-2xl text-white">\n                {sessionData &&
<span>Logged in as {sessionData.user?.name}</span>}\n
{secretMessage && <span> - {secretMessage}</span>}\n            </p>\n
 <button\n                 className="rounded-full bg-white/10 px-10 py-3
font-semibold text-white no-underline transition hover:bg-white/20"\n
    onClick={sessionData ? () => signOut() : () => signIn()}\n            >\n
            {sessionData ? "Sign out" : "Sign in"}\n            </button>\n
  </div>\n    );\n};\n
```

END index.tsx

START login.tsx

```
import { useRouter } from "next/router";\nimport { useEffect } from
"react";\nimport Spinner from "@components/spinner/spinner";\nimport {
useSession, signIn } from "next-auth/react";\n\nconst Login = () => {\n    const
router = useRouter();\n    const { status } = useSession();\n\n    useEffect(()
=> {\n        const run = async () => {\n            if (status === "loading")
return;\n            if (status === "authenticated") {\n
router.push("/dashboard");\n                return;\n            }\n
signIn("github");\n        };\n        run();\n    }, [status, router]);\n\n
return (\n        <main className="bg-slate-900">\n            <div
className="flex h-screen w-screen items-center justify-center">\n
<Spinner />\n            </div>\n        </main>\n    );\n};\n\nexport default
Login;\n
```

END login.tsx

START _app.tsx

```
import { type AppType } from "next/app";\nimport { type Session } from
"next-auth";\nimport { SessionProvider } from "next-auth/react";\nimport {
ChakraProvider } from "@chakra-ui/react";\n\nimport { trpc } from
"../utils/trpc";\n\nimport "../styles/globals.css";\n\nconst MyApp: AppType<{
session: Session | null }> = ({ Component, pageProps: { session, ...pageProps }
}) => {\n    return (\n        <ChakraProvider>\n            <SessionProvider
```

```
session={session}>\n                <Component {...pageProps} />\n
</SessionProvider>\n        </ChakraProvider>\n    );\n};\n\nexport default
trpc.withTRPC(MyApp);\n
```

END _app.tsx

START _document.tsx

```
// pages/_document.js\n\nimport { ColorModeScript } from
"@chakra-ui/react";\nimport { Html, Head, Main, NextScript } from
"next/document";\n\nexport default function Document() {\n    return (\n
<Html lang="en">\n            <Head />\n            <body>\n
<ColorModeScript initialColorMode={"dark"} />\n                <Main />\n
        <NextScript />\n            </body>\n        </Html>\n    );\n}\n
```

END _document.tsx

START _error.tsx

```
/**\n * NOTE: This requires `@sentry/nextjs` version 7.3.0 or higher.\n *\n *
This page is loaded by Nextjs:\n *  - on the server, when data-fetching methods
throw or reject\n *  - on the client, when `getInitialProps` throws or rejects\n
*  - on the client, when a React lifecycle method throws or rejects, and it's\n
*    caught by the built-in Nextjs error boundary\n *\n * See:\n *  -
https://nextjs.org/docs/basic-features/data-fetching/overview\n *  -
https://nextjs.org/docs/api-reference/data-fetching/get-initial-props\n *  -
https://reactjs.org/docs/error-boundaries.html\n */\n\nimport * as Sentry from
"@sentry/nextjs";\nimport type { NextPage } from "next";\nimport type {
ErrorProps } from "next/error";\nimport NextErrorComponent from
"next/error";\n\nconst CustomErrorComponent: NextPage<ErrorProps> = props => {\n
 // If you're using a Nextjs version prior to 12.2.1, uncomment this to\n  //
compensate for https://github.com/vercel/next.js/issues/8592\n  //
Sentry.captureUnderscoreErrorException(props);\n\n  return <NextErrorComponent
statusCode={props.statusCode} />;\n};\n\nCustomErrorComponent.getInitialProps =
async contextData => {\n  // In case this is running in a serverless function,
await this in order to give Sentry\n  // time to send the error before the
lambda exits\n  await Sentry.captureUnderscoreErrorException(contextData);\n\n
// This will contain the status code of the response\n  return
NextErrorComponent.getInitialProps(contextData);\n};\n\nexport default
CustomErrorComponent;
```

END _error.tsx

START users.tsx

```
import Layout from "@components/layout";\nimport Spinner from
"@components/spinner/spinner";\nimport { createColumnHelper, getCoreRowModel,
useReactTable } from "@tanstack/react-table";\nimport type { User } from
"@prisma/client";\nimport useGetUsers from "@hooks/useGetUser";\nimport Table
from "@components/table";\n\nconst columnHelper =
createColumnHelper<User>();\n\nconst columns = [\n
columnHelper.accessor("name", {\n        header: () => <span>Name</span>,\n
  cell: (info) => <i>{info.getValue()}</i>,\n    }),\n
columnHelper.accessor("email", {\n        header: () => <span>Email</span>,\n
    cell: (info) => <i>{info.getValue()}</i>,\n    }),\n
columnHelper.accessor("emailVerified", {\n        header: () =>
<span>Verified</span>,\n        cell: (info) =>
<i>{info.getValue()?.toLocaleTimeString()}</i>,\n    }),\n];\n\nconst Users = ()
=> {\n    const { data, isLoading } = useGetUsers();\n    const table =
useReactTable({\n        data,\n        columns,\n        getCoreRowModel:
getCoreRowModel(),\n    });\n\n    return (\n        <Layout>\n
<h1>Users</h1>\n            {isLoading && <Spinner className="flex w-full
justify-center" />}\n            <Table table={table} />\n        </Layout>\n
);\n};\n\nexport default Users;\n
```

END users.tsx

START assignments.tsx

```
import Layout from "@components/layout";\nimport Spinner from
"@components/spinner/spinner";\nimport useCourseFromRoute from
"@hooks/useCourseFromRoute";\n\nconst Assignments = () => {\n    const { data,
isLoading } = useCourseFromRoute();\n\n    return (\n        <Layout>\n
  {!isLoading && data && <>ID: {data.id}</>}\n           {!isLoading && !data
&& <>Course not found</>}\n           {isLoading && <Spinner className="flex
w-full justify-center" />}\n        </Layout>\n    );\n};\n\nexport default
Assignments;\n
```

END assignments.tsx

START index.tsx

END index.tsx

START index.tsx

END index.tsx

START [...nextauth].ts

```
import NextAuth, { type NextAuthOptions } from "next-auth";\nimport
GitHubProvider from "next-auth/providers/github";\n// Prisma adapter for
NextAuth, optional and can be removed\nimport { PrismaAdapter } from
"@next-auth/prisma-adapter";\nimport { prisma } from
"../../../server/db/client";\nimport { env } from
"../../../env/server.mjs";\n\nexport const authOptions: NextAuthOptions = {\n
pages: {\n        signIn: "/login",\n    },\n\n    // Include user.id on
session\n    callbacks: {\n        session({ session, user }) {\n            if
(session.user) {\n                session.user.id = user.id;\n            }\n
        return session;\n        },\n    },\n    // Configure one or more
authentication providers\n    adapter: PrismaAdapter(prisma),\n    providers:
[\n        GitHubProvider({\n            clientId: env.GITHUB_ID,\n
clientSecret: env.GITHUB_SECRET,\n        }),\n    ],\n};\n\nexport default
NextAuth(authOptions);\n
```

END [...nextauth].ts

START [trpc].ts

```
import { createNextApiHandler } from "@trpc/server/adapters/next";\n\nimport {
env } from "../../../env/server.mjs";\nimport { createContext } from
"../../../server/trpc/context";\nimport { appRouter } from
"../../../server/trpc/router/_app";\n\n// export API handler\nexport default
createNextApiHandler({\n    router: appRouter,\n    createContext,\n
onError:\n        env.NODE_ENV === "development"\n            ? ({ path, error
}) => {\n                console.error(`\u274c tRPC failed on ${path}:
${error}`);\n            }\n            : undefined,\n});\n
```

END [trpc].ts

START index.tsx

```
import { useEffect, useMemo, useRef, useState, useCallback } from
"react";\nimport { Box, Divider, Heading } from "@chakra-ui/react";\nimport
Layout from "@components/layout";\nimport Spinner from
"@components/spinner/spinner";\nimport AutoTable from
"@components/autoTable";\nimport useBreadcrumb from
"@hooks/useBreadcrumb";\nimport useCourseFromRoute from
```

```
"@hooks/useCourseFromRoute";\nimport type { Assignment } from
"@prisma/client";\nimport { trpc } from "@utils/trpc";\nimport type {
SortingState } from "@tanstack/react-table";\nimport { createColumnHelper } from
"@tanstack/react-table";\nimport { useReactTable, getCoreRowModel,
getSortedRowModel } from "@tanstack/react-table";\nimport Link from
"next/link";\n\nconst columnHelper = createColumnHelper<Assignment>();\n\nconst
columns = [\n    columnHelper.accessor("name", {\n        cell: (info) => (\n
        <Link
href={`/course/${info.row.original.courseId}/assignment/${info.row.original.id}`
}>\n                {info.getValue()}\n            </Link>\n        ),\n
header: () => <span>Name</span>,\n    }),\n    columnHelper.accessor("dueDate",
{\n        cell: (info) => info.getValue().toDateString(),\n        header: ()
=> <span>Due</span>,\n    }),\n];\n\nconst Course = () => {\n    useBreadcrumb({
href: "/courses", name: "Courses" });\n    const tableContainerRef =
useRef<HTMLDivElement>(null);\n    const [sorting, setSorting] =
useState<SortingState>([]);\n\n    const { data: courseData, isLoading:
isCourseLoading } = useCourseFromRoute();\n    const { data, isFetching,
fetchNextPage } = trpc.courses.assignments.useInfiniteQuery(\n        {
courseId: courseData?.id ?? "" },\n        {\n            enabled:
!!courseData?.id,\n            getNextPageParam: (_lastGroup, groups) =>
_lastGroup.nextCursor,\n            keepPreviousData: true,\n
refetchOnWindowFocus: false,\n        }\n    );\n\n    const flatData =
useMemo(() => data?.pages?.flatMap((page) => page.items) ?? [], [data]);\n
const totalDBRows = data?.pages?.[0]?.meta?.totalDBRows ?? 0;\n    const
totalFetched = flatData.length;\n\n    //called on scroll and possibly on mount
to fetch more data as the user scrolls and reaches bottom of table\n    const
fetchMoreOnBottomReached = useCallback(\n        (containerRefElement?:
HTMLDivElement | null) => {\n            console.log("scrolled");\n
if (containerRefElement) {\n                const { scrollHeight, scrollTop,
clientHeight } = containerRefElement;\n                //once the user has
scrolled within 300px of the bottom of the table, fetch more data if there is
any\n                if (scrollHeight - scrollTop - clientHeight < 300 &&
!isFetching && totalFetched < totalDBRows) {\n                    void
fetchNextPage();\n                }\n            }\n        },\n
[fetchNextPage, isFetching, totalFetched, totalDBRows]\n    );\n\n    //a check
on mount and after a fetch to see if the table is already scrolled to the bottom
and immediately needs to fetch more data\n    useEffect(() => {\n
fetchMoreOnBottomReached(tableContainerRef.current);\n    },
[fetchMoreOnBottomReached]);\n\n    const table = useReactTable({\n        data:
flatData,\n        columns,\n        state: {\n            sorting,\n
},\n        onSortingChange: setSorting,\n        getCoreRowModel:
getCoreRowModel(),\n        getSortedRowModel: getSortedRowModel(),\n
debugTable: true,\n    });\n\n    if (isCourseLoading) {\n        return
<Spinner fullScreen />;\n    }\n\n    if (!courseData) {\n        return (\n
       <Layout>\n                <Box>Course not found</Box>\n
</Layout>\n        );\n    }\n\n    return (\n        <Layout>\n
<Box>{courseData.name}</Box>\n            <Divider />\n
<Heading>Assignments</Heading>\n            <div\n                onScroll={(e)
=> fetchMoreOnBottomReached(e.target as HTMLDivElement)}\n
ref={tableContainerRef}\n                id="cuck"\n
className="h-[500px] max-w-[900px] overflow-auto"\n            >\n
 <AutoTable refReq={tableContainerRef} table={table} isFetching={isFetching}
/>\n            </div>\n        </Layout>\n    );\n};\n\nexport default
Course;\n
```

END index.tsx

START [assignmentId].tsx

```
import Layout from "@components/layout";\nimport Spinner from
"@components/spinner/spinner";\nimport useBreadcrumb from
"@hooks/useBreadcrumb";\nimport useCourseFromRoute from
"@hooks/useCourseFromRoute";\nimport useRouteParam from
"@hooks/useRouteParam";\n\nconst Assignments = () => {\n    const courseId =
useRouteParam("id");\n    useBreadcrumb({ href: `/${courseId}`, name: "courseId"
});\n    const { data, isLoading } = useCourseFromRoute();\n\n    return (\n
   <Layout>\n            {!isLoading && data && <>ID: {data.id}</>}\n
{!isLoading && !data && <>Course not found</>}\n            {isLoading &&
```

```
<Spinner className="flex w-full justify-center" />}\n        </Layout>\n
);\n};\n\nexport default Assignments;\n
```

END [assignmentId].tsx

START courseFilterSchema.ts

```
import { z } from "zod";\n\nexport const courseFilterSchema = z.object({\n
all: z.boolean(),\n    amount: z.number().min(1).max(100),\n});\n\nexport type
CourseFilter = z.infer<typeof courseFilterSchema>;\n
```

END courseFilterSchema.ts

START get-server-auth-session.ts

```
import { type GetServerSidePropsContext } from "next";\nimport {
unstable_getServerSession } from "next-auth";\n\nimport { authOptions } from
"../../pages/api/auth/[...nextauth]";\n\n/**\n * Wrapper for
unstable_getServerSession https://next-auth.js.org/configuration/nextjs\n * See
example usage in trpc createContext or the restricted API route\n */\nexport
const getServerAuthSession = async (ctx: {\n    req:
GetServerSidePropsContext["req"];\n    res:
GetServerSidePropsContext["res"];\n}) => {\n    return await
unstable_getServerSession(ctx.req, ctx.res, authOptions);\n};\n
```

END get-server-auth-session.ts

START client.ts

```
import { PrismaClient } from "@prisma/client";\n\nimport { env } from
"../../env/server.mjs";\n\ndeclare global {\n    // eslint-disable-next-line
no-var\n    var prisma: PrismaClient | undefined;\n}\n\nexport const prisma =\n
  global.prisma ||\n    new PrismaClient({\n        log: env.NODE_ENV ===
"development" ? ["query", "error", "warn"] : ["error"],\n    });\n\nif
(env.NODE_ENV !== "production") {\n    global.prisma = prisma;\n}\n
```

END client.ts

START context.ts

```
import { type inferAsyncReturnType } from "@trpc/server";\nimport { type
CreateNextContextOptions } from "@trpc/server/adapters/next";\nimport { type
Session } from "next-auth";\n\nimport { getServerAuthSession } from
"../common/get-server-auth-session";\nimport { prisma } from
"../db/client";\n\ntype CreateContextOptions = {\n    session: Session |
null;\n};\n\n/** Use this helper for:\n * - testing, so we dont have to mock
Next.js' req/res\n * - trpc's `createSSGHelpers` where we don't have req/res\n *
@see https://create.t3.gg/en/usage/trpc#-servertrpccontextts\n **/\nexport const
createContextInner = async (opts: CreateContextOptions) => {\n    return {\n
   session: opts.session,\n        prisma,\n    };\n};\n\n/**\n * This is the
actual context you'll use in your router\n * @link
https://trpc.io/docs/context\n **/\nexport const createContext = async (opts:
CreateNextContextOptions) => {\n    const { req, res } = opts;\n\n    // Get the
session from the server using the unstable_getServerSession wrapper function\n
 const session = await getServerAuthSession({ req, res });\n\n    return await
createContextInner({\n        session,\n    });\n};\n\nexport type Context =
inferAsyncReturnType<typeof createContext>;\n
```

END context.ts

START trpc.ts

```
import { initTRPC, TRPCError } from "@trpc/server";\nimport superjson from
"superjson";\n\nimport { type Context } from "./context";\n\nconst t =
initTRPC.context<Context>().create({\n    transformer: superjson,\n
errorFormatter({ shape }) {\n        return shape;\n    },\n});\n\nexport const
```

```
router = t.router;\n\n/**\n * Unprotected procedure\n **/\nexport const
publicProcedure = t.procedure;\n\n/**\n * Reusable middleware to ensure\n *
users are logged in\n */\nconst isAuthed = t.middleware(({ ctx, next }) => {\n
 if (!ctx.session || !ctx.session.user) {\n        throw new TRPCError({ code:
"UNAUTHORIZED" });\n    }\n    return next({\n        ctx: {\n            //
infers the `session` as non-nullable\n            session: { ...ctx.session,
user: ctx.session.user },\n        },\n    });\n});\n\n/**\n * Protected
procedure\n **/\nexport const protectedProcedure = t.procedure.use(isAuthed);\n
```

END trpc.ts

START auth.ts

```
import { router, publicProcedure, protectedProcedure } from "../trpc";\n\nexport
const authRouter = router({\n    getSession: publicProcedure.query(({ ctx }) =>
{\n        return ctx.session;\n    }),\n    getSecretMessage:
protectedProcedure.query(() => {\n        return "you can now see this secret
message!";\n    }),\n});\n
```

END auth.ts

START courses.ts

```
import { z } from "zod";\nimport { courseFilterSchema } from
"../../../schema/courseFilterSchema";\nimport ifTrue from
"../../../utils/conditioanl";\nimport { router, protectedProcedure } from
"../trpc";\n\nexport const coursesRouter = router({\n    all:
protectedProcedure.query(({ ctx }) => {\n        return
ctx.prisma.course.findMany();\n    }),\n    get:
protectedProcedure.input(z.string()).query(({ ctx, input }) => {\n        return
ctx.prisma.course.findFirst({\n            where: {\n                id: {\n
            equals: input,\n                },\n            },\n        });\n
   }),\n    assignments: protectedProcedure\n        .input(z.object({ courseId:
z.string(), cursor: z.string().nullish() }))\n        .query(async ({ ctx, input
}) => {\n            const limit = 50;\n            const { cursor, courseId } =
input;\n            const items = await ctx.prisma.assignment.findMany({\n
        where: {\n                courseId: courseId,\n                },\n
            orderBy: {\n                dueDate: "desc",\n
},\n                take: limit + 1,\n                cursor: cursor ? { id:
cursor } : undefined,\n            });\n            const totalDBRows = await
ctx.prisma.assignment.count({\n                where: {\n
courseId: courseId,\n                },\n            });\n\n            let
nextCursor: typeof cursor | undefined = undefined;\n            if (items.length
> limit) {\n                const nextItem = items.pop();\n
nextCursor = nextItem?.id;\n            }\n            return {\n
items,\n                nextCursor,\n                meta: {\n
 totalDBRows,\n                },\n            };\n        }),\n    active:
protectedProcedure.query(({ ctx }) => {\n        return
ctx.prisma.course.findMany({\n            where: {\n                published: {
equals: true },\n            },\n        });\n    }),\n    enrolled:
protectedProcedure.input(courseFilterSchema).query(({ ctx, input }) => {\n
 return ctx.prisma.course.findMany({\n            take: input.amount,\n
  where: {\n                enrolledUsers: {\n                    some: {\n
                user: {\n                    id: {\n
            equals: ctx.session.user.id,\n                    },\n
                },\n                },\n            },\n
 ...ifTrue(\n                input.all,\n                {\n
          published: {\n                    equals: true,\n
          },\n                },\n                {}\n
),\n            },\n        });\n    }),\n    enroll: protectedProcedure\n
 .input(\n            z.object({\n                userId:
z.string().nullish(),\n                courseId: z.string(),\n            })\n
    )\n        .mutation(({ ctx, input }) => {\n            const userId =
input.userId ?? ctx.session.user.id;\n            return
ctx.prisma.courseUser.create({\n                data: {\n
courseId: input.courseId,\n                userId,\n            },\n
       });\n        }),\n    unenroll: protectedProcedure\n        .input(\n
```

```
       z.object({\n                userId: z.string().nullish(),\n
 courseId: z.string(),\n            })\n        )\n        .mutation(({ ctx,
input }) => {\n            const userId = input.userId ?? ctx.session.user.id;\n
           return ctx.prisma.courseUser.delete({\n                where: {\n
               courseId_userId: {\n                            courseId:
input.courseId,\n                            userId,\n                    },\n
         },\n           });\n        }),\n});\n
```

END courses.ts

START user.ts

```
import { z } from "zod";\nimport { router, protectedProcedure } from
"../trpc";\n\nexport const userRouter = router({\n    all:
protectedProcedure.query(({ ctx }) => {\n        return
ctx.prisma.user.findMany();\n    }),\n    get:
protectedProcedure.input(z.string()).query(({ ctx, input }) => {\n        return
ctx.prisma.user.findFirst({\n            where: {\n                id: input,\n
          },\n        });\n    }),\n});\n
```

END user.ts

START _app.ts

```
import { router } from "../trpc";\nimport { authRouter } from "./auth";\nimport
{ coursesRouter } from "./courses";\nimport { userRouter } from
"./user";\n\nexport const appRouter = router({\n    auth: authRouter,\n
courses: coursesRouter,\n    users: userRouter,\n});\n\n// export type
definition of API\nexport type AppRouter = typeof appRouter;\n
```

END _app.ts

START globals.css

```
@tailwind base;\n@tailwind components;\n@tailwind utilities;\n
```

END globals.css

START next-auth.d.ts

```
import { type DefaultSession } from "next-auth";\n\ndeclare module "next-auth"
{\n    /**\n     * Returned by `useSession`, `getSession` and received as a prop
on the `SessionProvider` React Context\n     */\n    interface Session {\n
 user?: {\n            id: string;\n        } & DefaultSession["user"];\n
}\n}\n
```

END next-auth.d.ts

START conditioanl.ts

```
const ifTrue = <T>(condition: boolean, value: T, def: T) => {\n    return
condition ? value : def;\n};\n\nexport default ifTrue;\n
```

END conditioanl.ts

START stringFormat.ts

```
export const capital = (str: string | null | undefined) => {\n    return str ?
str.charAt(0).toUpperCase() + str.substring(1, str.length) : "";\n};\n\nexport
const capitalAll = (str: string | null | undefined) => {\n    return str ?
str.split(" ").map(capital).join(" ") : "";\n};\n
```

END stringFormat.ts

START trpc.ts

```
import { httpBatchLink, loggerLink } from "@trpc/client";\nimport {
createTRPCNext } from "@trpc/next";\nimport { type inferRouterInputs, type
inferRouterOutputs } from "@trpc/server";\nimport superjson from
"superjson";\n\nimport { type AppRouter } from
"../server/trpc/router/_app";\n\nconst getBaseUrl = () => {\n    if (typeof
window !== "undefined") return ""; // browser should use relative url\n    if
(process.env.VERCEL_URL) return `https://${process.env.VERCEL_URL}`; // SSR
should use vercel url\n    return `http://localhost:${process.env.PORT ??
3000}`; // dev SSR should use localhost\n};\n\nexport const trpc =
createTRPCNext<AppRouter>({\n    config() {\n        return {\n
transformer: superjson,\n            links: [\n                loggerLink({\n
                enabled: (opts) =>\n                        process.env.NODE_ENV
=== "development" ||\n                        (opts.direction === "down" &&
opts.result instanceof Error),\n                }),\n
httpBatchLink({\n                    url: `${getBaseUrl()}/api/trpc`,\n
      }),\n            ],\n        };\n    },\n    ssr: false,\n});\n\n/**\n *
Inference helper for inputs\n * @example type HelloInput =
RouterInputs['example']['hello']\n **/\nexport type RouterInputs =
inferRouterInputs<AppRouter>;\n/**\n * Inference helper for outputs\n * @example
type HelloOutput = RouterOutputs['example']['hello']\n **/\nexport type
RouterOutputs = inferRouterOutputs<AppRouter>;\n
```

END trpc.ts